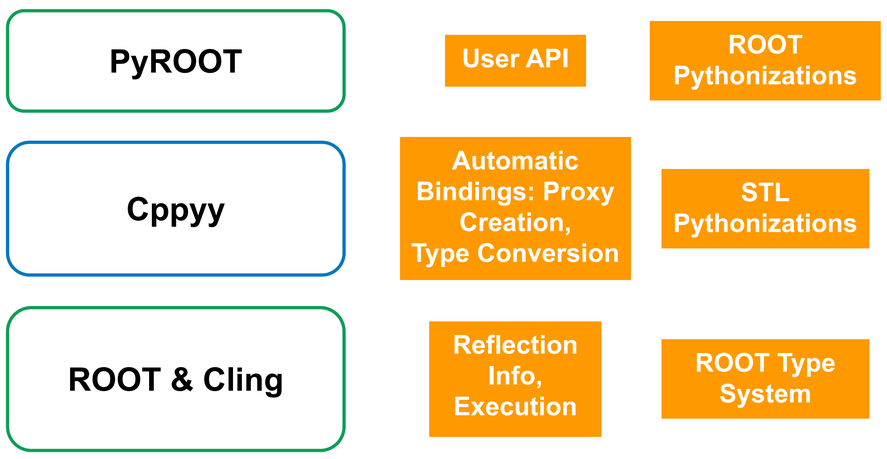
A New PyROOT: Modern, Interoperable and more Pythonic

https://root.cern

E. Tejedor, S. Wunsch, M. Galli
EP-SFT CERN

Modern

- **New design** on top of Cppyy libraries for automatic binding generation

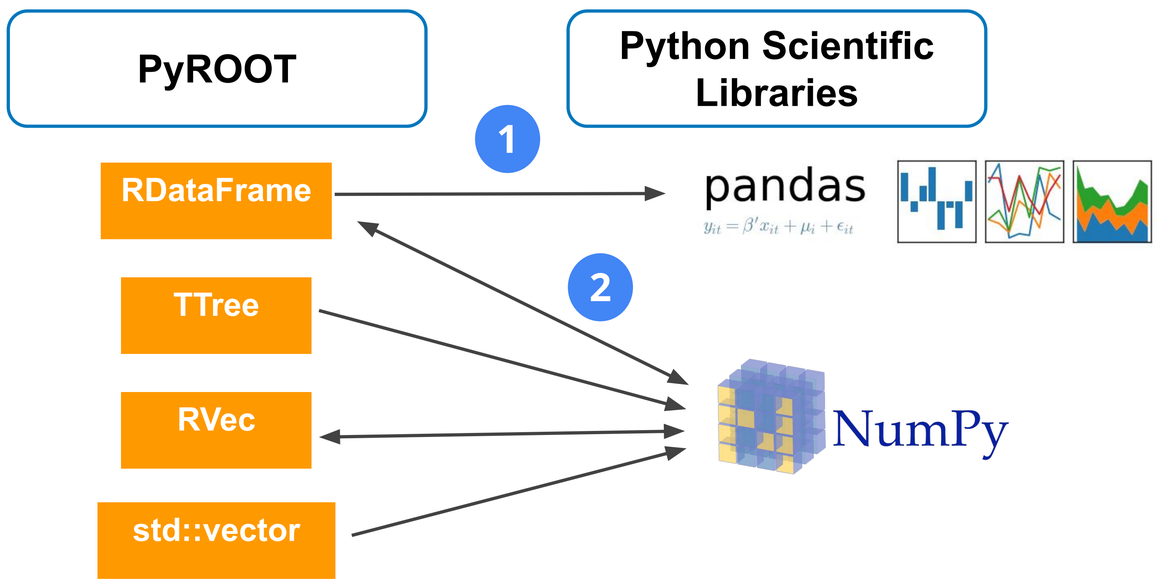


- Support for **modern C++** syntax

```
>>> import ROOT
>>> ROOT.gInterpreter.ProcessLine("""
template<typename... myTypes>
int f() { return sizeof...(myTypes); }
""")
0L
>>> ROOT.f['int', 'double', 'void*']()
3
```

Interoperable

- Integration with **Python data science** ecosystem




```
# Run input pipeline with C++ performance that can process TBs of data
df = ROOT.RDataFrame('tree', 'file.root')
     .Filter('pT_j0>30', 'Trigger requirement')
     .Filter('n_jet >= 2', 'Jet multiplicity cut')
     .Define('r_j0', 'sqrt(eta_j0*eta_j0 + phi_j0*phi_j0)')

# Read out final selection with defined variables as NumPy arrays
col_dict = df.AsNumpy(['r_j0', 'eta_j0', 'phi_j0'])   2
print(col_dict)

{'r_j0': ndarray([0.26,1.,4.45]), 'eta_j0': ndarray(0.1,-1.,2.1),
'phi_j0': ndarray([-0.5,0.,0.2])}

# Wrap data with pandas
import pandas
p = pandas.DataFrame(col_dict)   1
print(p)

   r_j0  eta_j0  phi_j0
0  0.26   0.1    -0.5
1  1.0   -1.0     0.0
2  4.45   2.1     0.2
```

Pythonic

- More **pythonisations** for ROOT classes
  - Make it easier to use ROOT C++ functionality from Python
  - Promote the use of Python syntax

```
myfile.mytree
```
vs
```
myfile.GetObject('mytree')
```

- Soon: support pythonisations of **user classes**
  - Lazily executed

```
@pythonization('MyCppClass')
def my_pythonizor_function(klass):
   # Inject new behaviour in the class
   klass.__iter__ = ...
```

New Build & Install

- Support for **multi-version** builds
  - Generate PyROOT libraries for multiple Python versions

```
$ cmake -DPYTHON_EXECUTABLE=/usr/bin/python3.6 ../root
$ cmake -DPYTHON_EXECUTABLE=/usr/bin/python2.7 ../root
```

  - Switch between Python versions

```
$ ROOT_PYTHON_VERSION=3.6 source bin/thisroot.sh
$ ROOT_PYTHON_VERSION=2.7 source bin/thisroot.sh
```

- Installation on **Python directories**
  - E.g. `/usr/local/lib/pythonX.Y/site-packages`
  - No need to set PYTHONPATH!

C++ Callables

- Automatically wrap Python callables with C++ callables
- Uses **numba** to compile Python callables
- Usage example: RDataFrame jitted string parameters

```
@ROOT.DeclareCppCallable(['float'], 'float')
def myfunction(x):
    return x * x

ROOT.CppCallable.f(5.0)
# Returns 25.0

df = ROOT.RDataFrame('tree', 'file.root')
df2 = df.Define('x2', 'CppCallable::myfunction(x)')
# New column x2 is calculated by invoking myfunction on column x
```

- New PyROOT in experimental mode
  - **To build it**: `cmake -Dpyroot_experimental=ON`
- Goal: make new PyROOT the **default in 6.22**